

# Sparse multivariate factorization by mean of a few bivariate factorizations.

Bernard.Paris@univ-grenoble-alpes.fr

2016

## Abstract

We describe an algorithm to factor sparse multivariate polynomials using  $O(d)$  bivariate factorizations where  $d$  is the number of variables. This algorithm is implemented in the Giac/Xcas computer algebra system.

## 1 Introduction

To my knowledge, there are three classes of algorithms to factor multivariate polynomials over  $\mathbb{Z}$  :

- reduction to bivariate factorization and Hensel lifting (Von zur Gathen and Kaltofen [3], Bernardin and Monagan [1])
- evaluation of one variable at a sufficiently large  $z$ , factorization and reconstruction by writing coefficients in basis  $z$  with symmetric remainders (heuristic factorization)
- Kronecker-like substitution (replace one of the variable by another one to a sufficiently large power).

Bivariate factorization may be obtained by partial differential equation (Gao [2]) or interpolation or one of the previous method.

We present here a method that is adapted to sparse factors, where the previous method would require too many ressources. For example Hensel lifting does not work if a leading coefficient of the factors vanishes once evaluated to 0 at other variables. The usual trick to avoid this is to translate the origin, but this will densify the polynomial to be factored.

## 2 The algorithm

### 2.1 Main idea

Let  $n \geq 2$  and  $P(x, x_1, \dots, x_n)$  be a sparse square-free polynomial that we want to factor, assume that the factorization is :

$$P = P_1 \dots P_k$$

The basic idea is replace all variables  $x_1, \dots, x_n$  with  $t, t, \dots, t$  and factor the substituted bivariate polynomial  $P_{t, \dots, t}$ , then compare with the factorization of  $P_{t^2, t, \dots, t}$  (where  $x_1, \dots, x_n$  are substituted by  $t^2, t, t, \dots, t$  in  $P$ ) or with  $P_{t^3, t, \dots, t}$  or etc. If the factorization is sparse enough, there is a good chance that the factors will be similar (same number of monomials, same pattern in  $x$ , same value for the coefficients), and the monomial power differences in  $t$  will give us the  $x_1$  contribution to the monomials. Doing the same for  $x_2, \dots, x_n$  will give us the reconstruction.

The details are a little more complicated, because we must take care of the content of the substituted polynomials  $P_{t, t, \dots, t}$  and of the order of the monomials having the same  $x$  powers in a given factor. The next example that motivated the implementation in Giac/Xcas will demonstrate the main idea, problems and solutions.

### 2.2 Example

The following example was discussed on the sage-devel list, it was obtained with a random generation command returning 2 polynomials in 5 variables. We make the product and try to factor it back. It was not factored by Sage 7.4 (with Singular 4 inside), but was reported to be factored by magma in 3s.

```
A:=3732480000000000*a^25*b^9*c^25*d^21*e^21 +
1866240000000000*a^20*b^9*c^25*d^24*e^21 +
3732480000000000*a^20*b^4*c^25*d^22*e^21 +
1244160000000000*a^20*b^4*c^28*d^21*e^18 +
3732480000000*a^16*b^4*c^25*d^21*e^20 +
1866240000000*a^16*b^4*c^26*d^21*e^18 +
1866240000000*a^13*b^6*c^25*d^21*e^17 +
12441600000*a^13*b^5*c^25*d^21*e^12 +
3110400000*a^13*b^7*c^23*d^16*e^12 +
12441600000*a^13*b^4*c^25*d^16*e^13 +
3110400000*a^16*b^4*c^20*d^16*e^12 +
622080000*a^13*b*c^21*d^16*e^12 +
233280000*a^13*b*c^20*d^17*e^8 + 77760000*a^13*b*c^15*d^18*e^8 +
25920000*a^13*b*c^15*d^14*e^10 + 25920000*a^13*b^4*c^15*d^10*e^8 +
17280000*a^8*b*c^15*d^14*e^8 + 3240000*a^8*b^4*c^15*d^6*e^8 +
216000*a^4*b^3*c^15*d^6*e^8 + 216000*a^4*b*c^10*d^9*e^8 +
86400*a^4*b*c^10*d^8*e^7 + 32400*a^7*b*c^10*d^3*e^7 +
2700*a^4*b^4*c^10*d^3*e^3 + 675*a^6*b*c^7*d^3*e^3 +
1125*a^5*b*c^2*d^3*e^3 + 135*b^5*c^2*d^3*e^3 +
```

$$27*c^2*d^6*E^3 + 12*c^7 + 9*c^3*d^3 + a^2;$$

[illegible]

The smallest partial degree of the product is  $9+19$  in  $b$ , therefore  $b$  will be our  $x$  variable, while  $a, c, d, E$  are our  $x_1, \dots, x_4$  variables.  $A$  and  $B$  are irreducible, we set  $P = AB$ .

Factoring  $P(x, t, t, t, t)$  gives

$$\begin{aligned} & t^5 * \\ & (3732480000000000 * b^9 * t^{90} + 1866240000000000 * b^9 * t^{88} \\ & + 31104000000 * b^7 * t^{62} + 186624000000 * b^6 * t^{74} + \\ & 124416000000 * b^5 * t^{69} + 135 * b^5 * t^6 + \\ & 3732480000000000 * b^4 * t^{86} + 124416000000000 * b^4 * t^{85} + \\ & 3732480000000000 * b^4 * t^{80} + 18662400000000 * b^4 * t^{79} \\ & + 124416000000 * b^4 * t^{65} + 31104000000 * b^4 * t^{62} + 25920000 * b^4 * t^{44} \\ & + 3240000 * b^4 * t^{35} + 2700 * b^4 * t^{18} + \\ & 216000 * b^3 * t^{31} + \\ & 622080000 * b * t^{60} + 233280000 * b * t^{56} + 77760000 * b * t^{52} \\ & + 25920000 * b * t^{50} + 17280000 * b * t^{43} + 216000 * b * t^{29} \\ & + 86400 * b * t^{27} + 32400 * b * t^{25} + 675 * b * t^{17} + 1125 * b * t^{11} \\ & + 27 * t^9 + 12 * t^5 + 9 * t^4 + 1) * \dots \end{aligned}$$

Factoring  $P(x, t^2, t, t, t)$  gives

$$t^7*$$

It is clearly a similar factorization, the number of monomials differ only by 1 ( $12441600000 * b^4 * t^{65} + 3110400000 * b^4 * t^{62}$  is grouped in one monomial in the second factorization), and the order is not the same for the coefficient of  $b$ . Note that there is also a content term in  $t$ . In fact, we just got an unlucky evaluation at  $x_1 = a = t^2$ ,  $x_1 = a = t^3$  is also unlucky, while  $x_1 = a = t^4$  returns 30 monomials like  $t$ .

In order to compare the two factorizations, we must solve these 3 problems: content, number of monomials, and monomials ordering.

We assume that the factors of the bivariate factorization are  $x$ -distincts, that is the distribution of non-zero coefficients in powers of  $x$  are not the same. This way, we can isolate the same factor in two bivariate factorizations, and we will now reconstruct the true factor.

4

$5t^3)/(t^{90} + 5t^{88}) = t^{63}$ . For  $P_{t^4, \dots, t}$ , we multiply the factor by  $t^{4 \times 22 + 31 + 54 + 41}(t^{20} + 5t^3)/(t^{161} + 5t^{144}) = t^{73}$ .

Solving the number of monomials is done like for any modular reconstruction: if an evaluation with  $x_k$  replaced by  $t^j$  contains less term than the previous one, we ignore it (unlucky evaluation), if an evaluation contains more terms than a previous one, we throw what we had before and restart from this one. If the number of monomials is the same, we also check that the non-zero partials degrees in  $x$  are the same.

Keeping the right order of monomials is more original: it can be done by comparing first the  $x$  power, then comparing the coefficient of the monomial (it is impossible to insure the same ordering by sorting with powers of  $t$ ). In order to do that we must insure that in the factor to be reconstructed the coefficients of the monomials of the same power of  $x$  are all distincts. If this is not the case, we can dilate some variables by a constant factor and retry (in our implementation we dilate all variables except  $x$  randomly by  $\pm 1$  or  $\pm 2$ ).

If we have two matching factors for evaluations at  $t..t, t^j, t..t$  and  $t..t, t^{j'}, t..t$  then the power of  $x_k$  in a monomial is the difference of powers in  $t$  of the same monomial in the two factorizations divided by  $j - j'$ . For example the first monomial in  $P_{t, \dots, t}$  multiplied by  $t^{63}$  is  $37324800000000 * b^9 * t^{90+63}$ , the corresponding monomial in  $P_{t^4, \dots, t}$  multiplied by  $t^{73}$  is  $37324800000000 * b^9 * t^{161+73}$ , that's a power contribution for  $x_1 = a$  of  $(234 - 153)/3 = 27$ . Indeed the leading coefficient of  $A$  is  $a^{25}$ , multiplied by  $a^2$  inside the leading coefficient of  $Q$  in  $b$  is  $a^{27}$ .

## 2.4 Implementation

This algorithm is implemented in C++ in the file `ezgcd.cc` of the source code of Giac/Xcas, in the function `try_sparse_factor_bi`. It factors the polynomial in the example in less than 2s (without this function, the factorization was impracticable).

We hope it will help other open-source softwares implement more efficient sparse multivariate factorization algorithms!

## References

- [1] L. Bernardin and M. B. Monagan. Efficient multivariate factorization over finite fields. In *International Symposium on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes*, pages 15–28. Springer, 1997.
- [2] S. Gao. Factoring multivariate polynomials via partial differential equations. *Mathematics of computation*, 72(242):801–822, 2003.
- [3] J. von zur Gathen and E. Kaltofen. Factoring sparse multivariate polynomials. *Journal of Computer and System Sciences*, 31(2):265–287, 1985.